APPLYING DEEP LEARNING FOR FOOD IMAGE ANALYSIS

MARC ASENJO PONCE DE LEÓN

Thesis Supervisors: Petia Radeva, Universitat de Barcelona & Computer Vision Center and Eduardo Aguillar, Universidad Católica del Norte, Antofagasta, Chile

A THESIS SUBMITTED FOR THE DEGREE OF MASTER IN ARTIFICIAL INTELLIGENCE

UNIVERSITAT POLITÈCNICA DE CATALUNYA, UNIVERSITAT DE BARCELONA & UNIVERSITAT ROVIRA I VIRGILI

April 2020

Abstract

With the increasing availability of data on the internet, deep learning techniques have been on the rise these past decade. Food images specifically are one of the most commonly shared types of image on social media. Because of this, the problem of food image analysis has been receiving increasing attention these past few years. However, it presents a series of challenges compared to other computer vision problems, which has limited the progress on the field. Nevertheless, some specific methods who capitalize on these challenges have been able to obtain good results.

In this thesis, the method of multi-scale multi-view feature aggregation (MSMVFA) applied to food recognition is explored. It is a strategy that has been able to obtain state-of-the-art performances on the literature recently. It capitalizes on merging information from different scales as well as different types, for instance ingredient and dish features. By using data coming from different granularity levels a more robust and discriminative classification is possible.

A detailed validation is provided to test the results of the method under different conditions. We see that both multi-scale and multi-view aspects of the strategy can be beneficial for the classification in various conditions for the food recognition problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer vision as a whole has evolved at a fast rate in the last 10 years, mainly because of the introduction of Convolutional Neural Networks (CNNs)[KSH17] in the field, which have been key to obtain the best results possible even today. Food image analysis is nothing but a small sub-space in computer vision, still recently its research has become very active because of its unique particularities (high intra-class variability and inter-class similarities, ambiguity, non-rigidness...). In this project, we will address the problem of food image analysis focusing on different methods that are currently being used in the field, specifically those that take advantage of these particularities of the problem. A lot of these methods are based on Deep Learning features as well as using the ingredient's information as a way to improve performance. Recently, it has been shown that to address this problem it is highly desirable to follow a Multi-Scale Multi-View Deep Feature Aggregation (MSMVFA) [JMLL19] approach. In MSMVFA, both multi-scale and multi-view features are aggregated, containing information both from dish and ingredient classification. Our project will specifically work on exploring in detail the multi-view multi-scale approach to find the best performance of food image analysis and recognition.

In the coming sections the overall Food Analysis field, applications and problems will be seen before getting into the current state-of-the-art from which this project develops.

1.1 Food Recognition Field

In Computer vision there is an entire field dedicated to object recognition. This one has been one of the fields with biggest growth in importance, both because of the challenges it poses as well as the endless applications it can provide to the world nowadays. With the huge increase in visual data availability (both images and video) has come an increase in method performances and also possible applications for them. Ranging from car detection to human pose analysis, the possibilities are limitless.

In this field of object recognition, there is a sub-field of food recognition, which has attracted increasing attention. This attention is due to the potential behind it. The increase in visual data has been reflected heavily in the subset of data of food. This can be seen from the following facts:

- There are 180 million photos with the hashtag #food on Instagram.
- 90 new photos hash-tagged #foodporn are uploaded to Instagram every minute.
- 54% of 18–24 year olds take a food photo while eating out.
- 39% of them have posted it somewhere online.
- 5% of over-50s share food snaps on online forums like Facebook and Twitter.

With the introduction of social media to our daily lives, the amount of food images that are taken and posted online has been increasing. It is clear that there is a potential in all these data, to be used in many ways. Various applications could be extracted from this, but here we see a summary of some of them:

- Health applications, such as being able to know the nutritional information of a meal so that a person can develop a better diet.
- Marketing applications, being able to analyze social trends from the correct detection of the images being posted online.
- Quality control, inspecting whether if a piece of food has been produced correctly.
- Daily applications, such as being able to keep a mobile visual food diary or having an automatic self-service in a restaurant.

As seen, these applications can range from corporate to personal use, and in some cases could have an important positive impact in people's lives. It is because of these factors that this field is considered to have a lot of potential. However, there is a reason why it has not reached the level in which it could be used for all these purposes, which we will see in the following section.

1.2 Food Recognition Challenges

As mentioned, food recognition has a lot of potential use-cases and applications nowadays. The reason why it has not been fully addressed yet is that it presents a series of challenges and problems compared to other traditional Object Recognition cases.

First and foremost, food categories have various geometrical variants, which can come from different viewpoints, scale or rotation. This means that using Convolutional Neural Networks

directly to extract visual features may fail if the variations are too large. This is portrayed in Figure 1.1. However, this is a common problem within many fields in object recognition. One thing that makes food stand out is the huge intra-class variations and inter-class similarities. As seen in Figure 1.2, images (a) and (b) are part of the same food category, but they have very distinctive appearances. This is an example of the big intra-class variability. However, image (c) is from another different category and, however, looks very similar to (a). This portrays the small inter-class variability. Nevertheless, as seen in the figure, extra information outside of the visual features like, for instance, the ingredients can help out in differentiating categories correctly. This is why many of the known datasets and methods include and take advantage of ingredient information [UB20a][UB20b][Gro20].

Figure 1.1: Three different geometrical shapes that some food categories can take

Figure 1.2: Image examples from the dataset VireoFood-172

Another special characteristic of food images is that they do not exhibit distinctive spatial layout and configuration. They are typically non-rigid, and the structure information can not be easily exploited. There are some methods for food recognition that are limited to food types like hamburgers, which have a distinctive spatial arrangement. This can be connected to the fact that, while being fine-grained recognition, semantic parts do not exist in many types of food. Semantic parts are usually exploited in object detection of birds or cars, in which you expect to see a beak and a wing or wheels and glasses. Many food categories however have an ambiguous definition. As an example, taking a look at image (b) in Figure 1.2 some people would say that it is a stuffed pineapple, others will say that it is rice with pineapple, others may say completely different things. There typically is no defined way that a specific food category looks like, which makes it harder to discriminate between them.

One of the main pillars of deep learning is data. Without enough of it, modern models would be unable to learn properly. Food recognition is no exception and, because of the reasons stated before, is usually even more dependent on the amount and quality of data. This presents a problem because there currently are no huge datasets available such as ImageNet for object recognition to help advance the development. Furthermore, many of the available data is poorly labeled, introducing extra noise to the learning procedure. In chapter 5 we will take a look at some of these available datasets, including those that have been relevant for the development of this project.

Chapter 2

Objectives

Working in such a wide field of research as is Food Image Analysis, there are many paths to chose from when developing a project like this one. It is chosen to prioritize the learning experience and developing something that can be of some usefulness rather than obtaining a marginal improvement on a method. This is why the main objectives of this project can be summarized as follows:

1. Researching and gaining a good understanding of the state of the art and different methods available in the field.
2. Exploring the Multi-view Multi-scale approach and the usefulness of the different methods used within this context.
3. Provide detailed validation to test the results under different conditions.

Overall, the main focus of this project, which are the end results that have been prioritized during the development, are objectives number 2 and 3. Applying obtained knowledge to be able to reproduce a complex method within the multi-scale multi-view framework and obtain meaningful results from which conclusions can be extracted is our main goal. A rigorous validation of the tests and results is key to this goal's completion as well.

Chapter 3

State of the Art

Computer Vision and Object Recognition as whole are fields that have been in active research for a long time, but specially recently. Deep Learning and the introduction of Convolutional Neural Networks have allowed these research spaces to rapidly advance, both of them being of big interest nowadays. Food Recognition is not an exception to this, and as such it has gained attention recently resulting on a lot of new research studies and proposals. Before this revolution, classic computer vision techniques were used, like the use of hand-crafted features based on descriptors like SIFT. In this section we will take a look at the overall state-of-the-art of Food Image Analysis, as well as some techniques commonly and recently used in this field.

3.1 Convolutional Neural Networks in Food Recognition

Just like with normal object recognition, CNN's are a very crucial part of most state-of-the-art methods for food recognition. The visual features obtained by these approaches have generally obtained better performance than the previous hand-crafted features because of the capacity of CNN's to learn representative and meaningful features. Some examples of these CNNs being applied to this field are the work of Yuzhen Lu [Lu16], in which a four-layer network is build (three convolutional-pooling layers and one fully-connected layer) to prove that deep learning techniques outperform traditional computer vision techniques, and the work of Kagaya et al. [KAO14], which adopts the well-known AlexNet [KSH17] to extract deep visual features for food recognition.

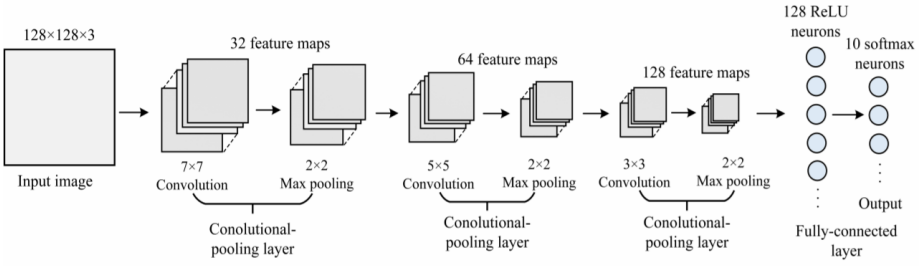

Figure 3.1: Schematic of a CNN used, extracted from [Lu16]

Further and more current studies also try to use additional external information to help on the process of food recognition. An example of this is Myers et al. [MJR+15] which uses the GoogLeNet network [SLJ+14] to recognize the content of the meal from an image, and includes GPS information in order to predict the nutritional content.

A very important aspect of food, however, is the added element of their ingredients as possible additions for the recognition process. As stated before, adding this information can greatly help in reducing the effect of the huge intra-class variability and small inter-class variability that food images present. Compared to normal food recognition, ingredient recognition is a multi-label problem, as multiple ingredients are present in the same image.

This is a problem that can also be tackled using CNN's in an effective manner. An example of this is the work from Zhou et al. [ZL16] which tried to take advantage of the relationship existing between ingredients, food category and restaurant information to use a bi-partite graph for food image classification. Another instance is the work of Min et al. [MJS+17] which does simultaneous ingredient and food recognition, using the ingredients as supervised information for fine-tuning the network. Other works concentrate solely on trying to detect ingredients of an image, even if they cannot be seen, by using known recipes [BFR17].

3.2 Ontology-driven methods

An ontology is a hierarchical structure that organizes data to represent knowledge like relationships between elements. These hierarchies can vary from trees to fully-connected graphs. The knowledge represented in them is obtained externally and independently from the images themselves. Ontologies can be applied in various ways in object recognition, for example using the ontology weights as regularization terms in objective functions. They can also be directly implemented and added into the model itself, like in the case of the work of Zhang et al. [ZQL+19] in which a two-layer ontology is built from the 19-layer WordNet (semantic ontology) or from a spectral clustering of mean deep visual features

(visual ontology), and then is added on top of Inception-V3.

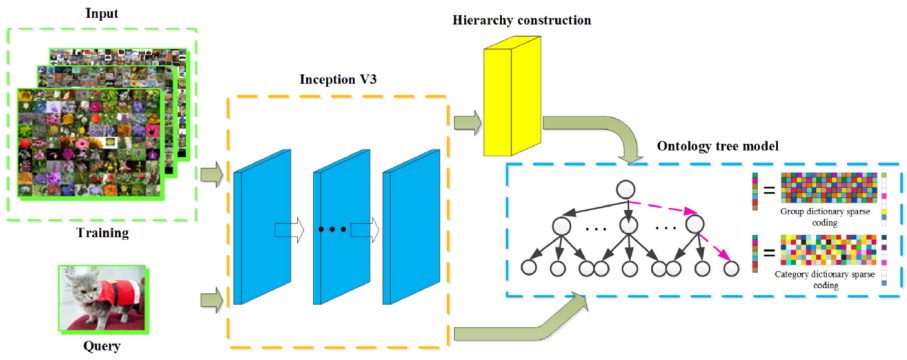

Figure 3.2: Example of the use of an ontology, extracted from [ZQL+19]

Sometimes, however, it's not the ontologies assisting on the learning process of a network, but it is the learned network the one that helps build the ontology. In the work of Zhang et al. [ZMZF19] a visual-semantic tree is learned in the visual space while considering inter-category semantic constraints to organize over 10000 image categories hierarchically, using deep visual features. This structure can then be used to classify images by adding either SVM classifiers or small CNN's to the structure's nodes. Another example of this type of methods is the work of Aditya et al. [AYB+18] in which visual detection, a knowledge base and logical reasoning is used to build a so-called Scene Description Graph. This ontology structure tries to combine all available information in the image and their relations.

There are many other works [ZSXL19][LFH16][CXH+19] which study the construction of different ontologies to aid on different types of problems. Food recognition can highly benefit from many of these methods because of the nature of food containing ingredients. Ontologies can be built to represent the relationship between different food categories and all possible ingredients, like the probability of inclusion in the recipe. This knowledge can then be implemented in all the possible ways we have seen in this section to improve the performance of the overall food recognition task.

3.3 Graph Convolutional Networks

Graph Neural Networks (GNNs) are a very recent field of research, which studies a generalization of classical NNs. They are based on a graph structure (usually fully-connected) in which each vertex has a hidden state that is updated at each step using a function that takes into account the input received in the vertex. There is a second function which is the one that computes this input from taking into account all the neighbours of the vertex and

the message passes through the connecting edges. It builds a message passing algorithm in which the functions used are specified by neural networks.

An example of recent uses of this kind of structure is the work of Durand et al. [DMM19] in which a GNN is built with a vertex per category, with initial hidden states being outputs of a CNN and using MLP and GRU respectively for the message update function and the hidden state update function. This method is used to model the correlation between the different categories, with the objective to work with partial labels and deduce missing labels. Another example is the work of Chen et al. [CWWG19] in which a GCN of several layers is used to apply successive convolutions to the hidden states of each vertex (representing each a category). The output of the final layer is considered in this case as a matrix of classifiers, which is used for a multi-label classification as seen in Figure 3.3. This structure of GCN allows for implicit and explicit relations between the nodes by sharing the weights and the gradients impacting one-another.

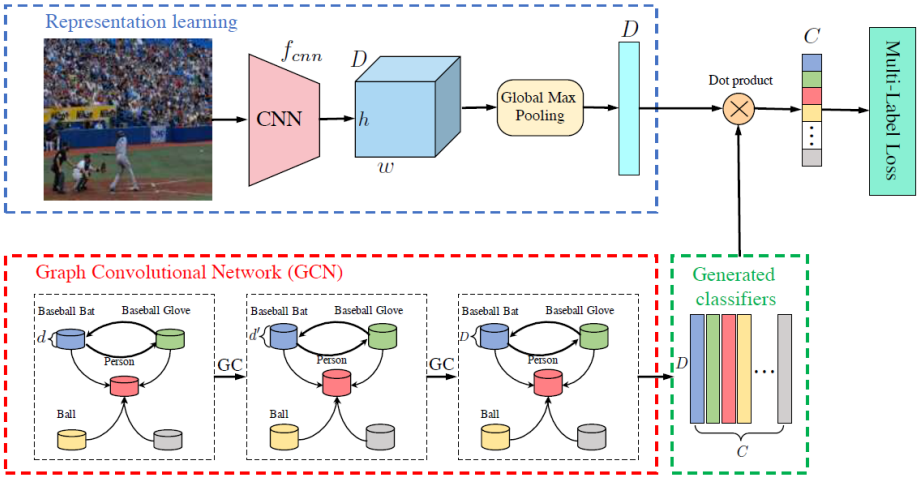

Figure 3.3: Example of the use of a GCN in the work [CWWG19]

Although there are not many works that specifically apply this method to food recognition in the moment, it is a powerful way to represent and exploit category interrelations. In the field of food recognition specifically, this could also be applied using ingredient information to model the interactions between the food categories. This could be seen as an alternative to ontology methods in the sense that it allows to apply additional knowledge to the final classification.

Chapter 4

Multi-Scale Multi-View approach for Food recognition

In this chapter we are going to take a look at the Multi-Scale Multi-View Feature Aggregation for food recognition [JMLL19].

From what we have seen in previous chapters, food recognition presents many particular characteristics compared to other fields in object recognition. Many of the ideas applied to this method come from trying to overcome some of these challenges. The first one is the multi-view aspect, which means that different types of feature sets are being aggregated. These different types being features with different granularity help in obtaining a better prediction and face the challenge of food categories having an ambiguous definition. Also the problem of high intra-class variability and small inter-class variability is tackled by this method because of the use of ingredients, as we will see. The second aspect of the method is the multi-scale part, which mainly denotes that features are obtained at different scales. One of the scales is the whole image to preserve spatial layout and the finer scales allow to capture more fine-grained details of the image. This allows the features to be more robust and invariable to the geometrical deformation that food images typically present. All features obtained using both of these approaches are finally joined together for a final classification step. An overview of the whole method can be seen in Figure 4.1, while we now go deeper into the different features and methods used.

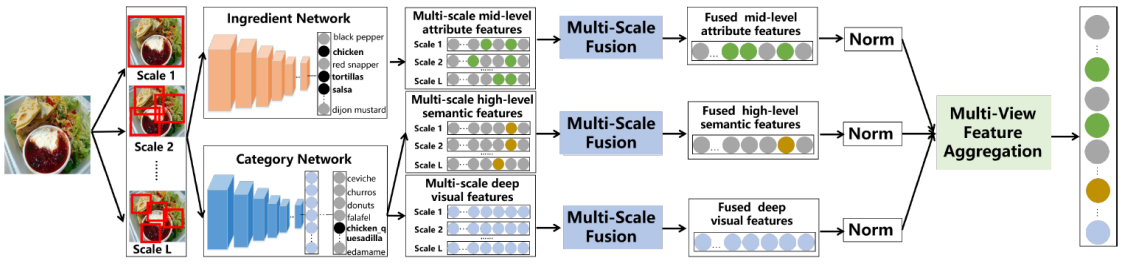

Figure 4.1: Overview of the MSMVFA method, (preprint [JMLL19])

4.1 Features

In this section we will take a look at the different features used. Starting with the multi-view part of it, there are a total of three different types of features used, which are the following:

- **Mid-level Attribute Representation:** The output of a trained model called Ingredient Network in Figure 4.1, which detects the present ingredients in the image. This model is trained as a multi-label classification network, and as such uses the sigmoid activation function in the last layer and the binary cross-entropy loss function. The result is a vector of features of equal size as the amount of different ingredients to be detected.
- **High-level Semantic Representation:** The output of a trained model called Category Network in Figure 4.1, which is trained to detect the food category in the image. As a single-label problem, this network will use the softmax activation function in the last layer and the categorical cross-entropy loss function. In the end, the output obtained is a vector of the same size as the amount of different food categories, which denotes a probability distribution.
- **Deep Visual Feature Representation:** Extracted from the last non-output layer of the Category Network, which contains class-relevant information that could prove useful. Its size will depend on the type of network used in the implementation.

Next, we will see the multi-scale part of the different features. For the full image, an Ingredient Network as well as a Category Network will be trained using full images. For each other scale of patches, however, a new Ingredient Network and Category Network will have to be trained using these patches. In test time features are extracted for all of the patches of that scale of the image, and then the features for all of them are fused into one, obtaining a vector of the corresponding size of the feature type.

4.2 Aggregation

Now that we know which kind of features are being used, we will see how the actual aggregation of all of them is done. First of all, features are obtained for all types and all scales. The first thing to do is the multi-scale aggregation, which will aggregate all scale features of the same type, for example the mid-level features (ingredient information). The aggregation process can be a simple concatenation, but before doing it the features coming from different scales should be normalized (z-score normalization in this case) so that values from all features are in the same ranges. This aggregation can be expressed as the following expression, with L_1, L_2 and L_3 being the features coming from 3 different scales, as an example:

$$Fn = Agg(Norm(L_1), Norm(L_2), Norm(L_3))$$

The following step is to aggregate all the features from a multi-view perspective. That is, the Mid-level features, High-level Features and Deep Visual Features that are already containing information from all different scales used. The procedure is the same as before, with even more importance on the previous normalization than before. This is because in this case features come from very different places and can have very different values depending on the type. The final expression of the multi-view aggregation looks like this, with F_1, F_2 and F_3 being the three feature aggregations obtained through the previous expression:

$$F = Agg(Norm(F_1), Norm(F_2), Norm(F_3))$$

This final feature vector F is the one that is going to be fed to a softmax classifier, which will be trained with the aggregated features to predict the correct food category in the end. In train time for each image all this process has to be done: obtain all feature representations, multi-scale aggregation for each type, multi-view aggregation of all feature types and run it through the softmax classifier to obtain a prediction. The exact same process has to be followed during test time.

An important consideration to make is which network architectures to use for both Ingredient and Category Networks. Usually, three options are studied: VGG-16 [SLJ+14], ResNet-152 [HZRS15] and DenseNet-161 [HLvdMW16]. The same architecture is always used for all of the networks present in the method, both for multi-scale and multi-view variations. In terms of multi-scale, three different scales are used: one for the whole image (L_1), one for the image divided in 4 patches (L_2) and the same but with 16 patches (L_3).

Chapter 5

Datasets

As stated previously, one of the particularities of the Food Analysis problem is that there are no huge datasets such as ImageNet or PlacesNet specifically built for this field. However, for the research to advance in the field datasets are mandatory, so there have to be some widely known and used ones. In this section we will take a look at the most popular datasets for the Food Recognition problem, some of which have also been used during the development of this project.

5.1 ETH Food-101

The Food-101 dataset [ETH20] [BGVG14] was built by researchers at ETH and is nowadays one of the most common datasets to use in Food Analysis research. As stated by its name, it contains a total of 101 dishes or food categories. There is a total of 101000 images, 1000 per category. Only training and test sets are specified in the dataset, with 750 images going to the training set and 250 going to the test set for each of the food categories. This is a challenging dataset that keeps noise both in the images and the true labels. Top-performing state-of-the-art methods obtain an accuracy of around 90% for this dataset.

In the development of this project, this has been the most important and most used dataset. It is the one for which the results are portrayed more extensively.

5.2 Recipes5k

Recipes5k [UB20b] [BFR17] is a dataset built by a research group at the Universitat de Barcelona (UB). It consists of 4826 unique recipes, which contain an image of a dish and the corresponding list of ingredients. In total, it contains 3213 unique ingredients, with an average of 10 of them per recipe. The dishes represented are those present in the ETH Food-101 dataset, so a total of 101 different dishes. The recipes represent alternative ways

to prepare the same dish. It also provides balanced training, validation and test splits.

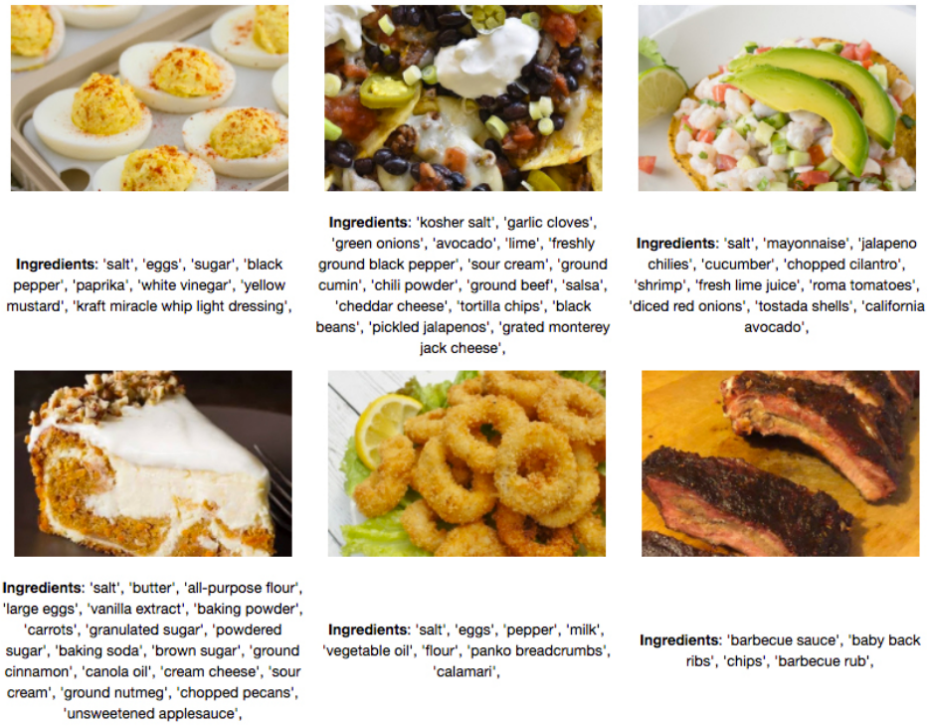

Figure 5.1: Examples of recipes in the Recipes5k dataset with their images and ingredients

This dataset has been used in the early stages of the project to build a toy problem to quickly test the results of the implementation.

5.3 Ingredients-101

Ingredients-101 [UB20a] [BFR17] is a dataset for ingredients recognition that is a result of the same work from which Recipes5k was created. The difference between both datasets is that Ingredients-101 does not contain any images as it consists of a list of the most common ingredients for each of the 101 food categories present in Food-101. In total, there are 446 unique ingredients in the main dataset, with an average of 9 per food category. However, a simplified version exists which only maintains 227 unique ingredients by further applying blacklisting. Although no images are included, the same train/validation/test split is proposed as in Recipes5k.

This dataset works as an extension of Food-101 for ingredients. For this reason it is used in the development of this project. More specifically, the simplified version that presents 227 ingredients is used in this case.

5.4 VireoFood-172

VireoFood-172 [Gro20] [JjC16] is a dish and ingredient dataset that contains 110241 images. There is a total of 172 food categories and 353 ingredients, both from Chinese cuisine. The number of images present for each category is not the same, so a random selection of a percentage of images has to be made from each category individually to make sure that the training/validation/test splits are done correctly. This split is not included in the dataset.

5.5 ChineseFoodNet

ChineseFoodNet [Mid20] [CZZD17] is a food category dataset with a total amount of 185628 images. These are divided throughout 208 Chinese food categories. It provides a dataset split of 145065, 20253, and 20310 images for training, validation and testing, respectively. However, the label information for the test set is not provided. Because of this some works prefer to re-define the split using, for example, the validation set partially for validation and testing. Like with Food-101, no ingredient information is provided.

Chapter 6

Validation

In this section, we will take a look at the development of the project itself, which includes the implementation of the MSMVFA method as a whole and testing its results with parameter variations as well. This experimental procedure can be divided into several stages, which are the different sections that can be found next, starting by the definitions of datasets and metrics used and finishing with the results obtained from the tests and a discussion of them.

The objective is to explore the performance of the MSMVFA on the Food-101 dataset. However, the first step necessary is to assure the correct implementation of the method and see how it works on a simple problem. For this reason, it was decided at first to do so using a toy problem with only 10 food categories, which was built from the Recipes5k [UB20b] dataset. This dataset's skeleton of data-processing was already available to use on Google Colab, which is the platform that was used for the implementation on this step [Ase20b][1]. This initial toy problem will be studied in this chapter.

The next step after the toy problem is to move to the Food-101 dataset so that the results can be compared with the SotA. In this chapter we will also see how this second stage of the project was developed. Several changes to the existing code had to be made, and new considerations had to be taken into account, as we will see. However, because of the results obtained during the first set of tests on Food-101, a series of changes had to be made. This is intended to be the final attempt at obtaining the same results as in the literature, so the methodology used has to be as similar as possible. Since the ingredient network was behaving in an unstable way, changes were made. Instead of using the developed code directly for the training of the category and ingredient networks, the use of a Multi-modal Keras Wrapper [Bol20] was introduced. We will see the details from this third stage of the project in the following sections as well.

[1] link to the colab: https://colab.research.google.com/drive/1aaL5xYmAGsY-j3TxkpgAsJHG6XF6rP1G

6.1 Implementation

6.1.1 Data processing

Recipes5k

From all the different food categories, the 10 selected for this stage of implementation were the following: *apple pie, carrot cake, cheesecake, chocolate mousse, tiramisu, panna cotta, waffles, red velvet cake, cupcakes, frozen yogurt.* All of them are desserts that could be difficult to differentiate depending on the image. With this being a toy problem, and because of the developing environment, in this stage all data is directly put in memory. Training, validation and test images and labels from the specified food categories are extracted from the dataset. This includes not only food category labels, but also ingredient labels for each image, which are encoded for a multi-label problem.

Both images and labels for the different scales have to be created in this step too. As we saw on the last chapter, it will be necessary to train an independent ingredient and category network for each of the scales. As this is a toy problem, it was initially decided to only take into account two different scales. These two scales are the following:

- L_1**:** Scale representing the whole image. The networks will be trained with the images themselves as they are.
- L_2**:** Scale represented by the images divided into 4 patches. In this case the networks will be trained with these image patches instead of the whole images.

For this reason, it is decided to create the image divisions beforehand and create a separate images array for the networks of L_2 instead of doing so during training time. This needs to be done for all images: training, validation and test splits all included. The image patches are then re-scaled to the same size as the original image, as the original authors specified. In Figure 6.1 an example of this image division can be seen.

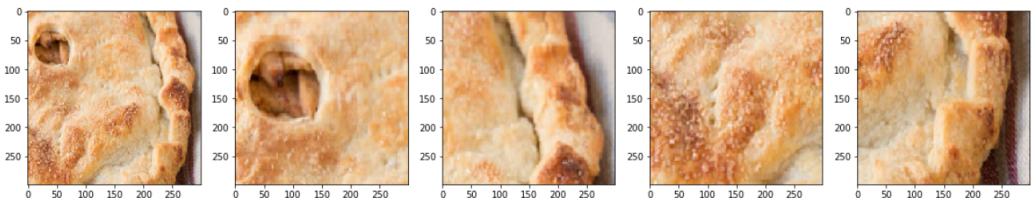

Figure 6.1: Example of image division: whole image of food category 'apple pie' on the left, its four divisions re-scaled on its right.

Considering training, validation and test partitions as well as the different scales, 6 different

image structures are obtained in the end, while a total of 12 structures for labels are necessary because of all images needing a category network label (food category) and an ingredient label (multi-label ingredients). Once all necessary image and label structures have been created, we can move on to the next step.

Food-101

Changing from one dataset to the other is not an easy task when the first one is just a toy problem. Moving to a much larger dataset brings new complications that need to be taken care of. Additionally, the execution setup for this stage of the project changes: that is, we move from Google Colab to a server-computer with more computational power. This overall will mean that a big part of the original code will have to be re-done and re-structured to adapt to the changes in the setup. We will see some of these adjustments now.

The implementation being done on a server, the code is now divided in several files that contain code for differentiated purposes. The final code obtained during the development of this project can be found on GitHub [Ase20a][2]. Organizing the code can be crucial when debugging it, specially when working with neural networks which can be hard to interpret.

One of the biggest changes, as mentioned before, is the size of the dataset used. We are now talking of 101000 images in total that need to be used. Not only that, but the images are larger in general compared to those on Recipes5k. This means that they can not be stored all at the same time in memory and, as such, a batch generation mechanism needs to be used, like `flow_from_directory` from Keras. This will make sure that memory does not run out when training if a correct batch size is used in the process. The dataset must be re-organized for this new method to be used compared to how it originally is downloaded.

Another consideration that comes to mind is what to do with smaller scales, like the case of L_2 from last stage. The solution is the following: images are processed independently before any training takes place and a new dataset is generated with the patches of images of the desired scale. This new dataset is also organized in a way that it can be used with a batch generation function. The same L_1 and L_2 scales are kept from the existing code, as well as the function which generates the patches and re-scales them to the original image size.

Having moved to a new datset which does not contain ingredient information, we have to make some adjustments. Just like the authors of the paper did, we use the simplified version

[2]Link to the code: https://github.com/Kittus/msmv

of Ingredients-101 to add the missing ingredient information to the dataset. Because of how Ingredients-101 is done, we can compute a conversion between food category and ingredient list. This is possible because the same ingredients are assigned to all images of the same category. This conversion is then applied during the batch generation to assign the correct labels to each image when training the ingredient networks.

6.1.2 Network training

Recipes5k

The next step is to train all the category and ingredient networks that will be used in the MSMVFA method. This could theoretically be done while also applying the MSMVFA method and training the final softmax classifier, but it would be very costly and hard to debug and maintain. A more step-by-step approach is taken in this implementation.

Although many parameters used for the training are specified in the original paper, there are other considerations that are not mentioned. Nevertheless, in this step which is only meant to check the correct implementation of the method as a whole, the specific accuracy obtained by the networks or their specific training procedure is not of much interest, specially since it is working on a different dataset. Because of this, the parameters used for this step are chosen by hand.

The first thing to consider is which network architecture is going to be used. From the different possible options, VGG-16 is selected for this task given the excellent results it obtained in other domains. The Keras VGG-16 model is used [Tea20] and it is loaded with pre-trained weights on ImageNet like the original authors did. The next question is whether if the fully-connected layers are taken as well or created anew, and whether if the full architecture is fine-tuned or only part of it is. For this particular problem many configurations where tried out, and in the end the architecture used is as follows:

- It does not include the FC layers of VGG-16, and adds one custom FC layer before the output layer.
- The last 4 layers (one convolutional block) of the VGG-16 network are also fine-tuned as the custom FC layer is trained.

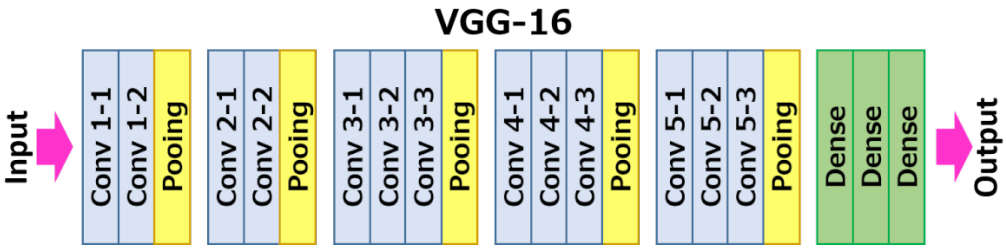

Figure 6.2: Scheme of the VGG-16 architecture used for all networks

In terms of the training procedure itself, Adam [KB14] is chosen as the optimizer. The number of epochs used is 30. The parameters that were toyed were the learning rate, the dropout added before the last layer and the amount of neurons in the custom FC layer. The chosen final parameters, as well as the results of the training of all these networks, will be shown and discussed in the following chapter. A total of 4 networks are trained in this step, belonging to the two scales and two types: ingredient and category. While the parameters chosen can be different between network types, they are kept the same between networks of the same type and different scale.

Food-101

In this subsection we will take a look at how the ingredient and category networks were trained for this second phase of the project. Having access to these trained models by the literature could prove useful to be able to more closely reproduce the author's results by only having to implement and train the MSMVFA method and the final classification. They could also be used just to check if this project's implementation and training of these models has been done correctly, by taking a look at the trained model's output, which is not specified in the original paper. Nevertheless, because this information is not available, the training of the networks will have to be done manually, which is what we will see in this section.

Compared to the experiments run with the Recipes5k dataset, we now do care about the output specifically and we want it to be as close as possible to the SoA results. As a result, we take a closer look at all the parameters for the training of the networks. The optimizer is changed from Adam to SGD with momentum. The dropout used in the previous networks is eliminated. However, some parameters are left with values that will allow the training to be faster for this stage, like the batch size. Another important change is that now the FCs from the VGG-16 pre-trained in ImageNet are included. This means that the input is restricted to have a size of (224, 224, 3), as seen in the Keras documentation [Tea20]. During the training process only these FCs are fine-tuned.

An extra implementation necessary at this point was to make the learning rate change after 10 epochs of training (out of 30 total). The MSMVFA method used to divide the learning rate by 10 and thus help the network find a better local minimum for better performance. This was achieved by using a custom callback in Keras.

All specific parameters and results will be seen next chapter. Improvements and changes were made later on, which we will see in the next section. Nevertheless, MSMVFA was also applied to the features obtained during this stage.

Because of the experience from the first reproduction of results and the drive to get as close as possible to the same methodology as the one used in the literature, many parameters were changed. These changes affect both the base networks as the softmax classifier, and can be summarized as follows:

- Weight decay was added to the category and ingredient networks (weight decay of 0.0001).
- The learning rate decay for both category and ingredient networks was fixed so that it is divided by 10 at epoch number 10 as wanted.
- The batch size for the training of the category and ingredient networks was set to 48, like in the original work.
- All layers are fine-tuned during the training of the category and ingredient networks, instead of only the two last layers. This comes with a pretty big cost in training time.
- The optimizer for the training of the softmax classifier is set to the same one as the base networks, SGD with momentum, with a set learning rate.

Parting from these initial changes, several configurations were later on tried out, as we will see. It is important to note that still the scales used to study are only L_1 and L_2. Although it would be preferable to also have the L_3 results to compare with the original results, it was not possible for time reasons. training one of the networks for L_1 can take up to 1 or 2 days, which is multiplied when doing it for L_2 because of these networks using 4 times as many images both for training as test sets. For L_3 the amount of images is multiplied even further, making it not viable for this project.

In this stage the Multi-modal Keras Wrapper is used for all ingredient and category network training procedures. This means having to set up the data in a specific way, specially the Ingredients-101 data to be integrated with Food-101. After the data has been processed and the model to be used is specified, it is a matter of changing parameters, which gives less

chance of error than using an original code.

Using this tool, several parameter modifications were made. Since using the same parameters as the authors had specified resulted in sub-optimal results, the decision was made to try and obtain the best performing networks. The variations were tested in a sequential manner, and always making a decision for the next taking into account past results. Here is a list of different changes made:

- With or without basic data augmentation.
- Changing from a Tensorflow optimizer to a Keras optimizer.
- Learning rate modifications ranging from 0.01 to 0.0001.

Once a good result was found independently for the ingredient and category networks for L_1, the same parameters were executed on their L_2 versions. Once all models were obtained, the rest of the process was left the same, keeping the softmax classifier with the literature's parameters. It is after this procedure that the best results have been obtained. We will see all these results and the reasoning behind them in the next chapter.

After the reproduction of results from the original paper was finished, the objective was to try to come up with ideas to improve the obtained results. One of the principal differences that even in the end kept existing between the implemented method and the one proposed by the authors was the image input size. While the authors used images of 256x256 all throughout their network training procedures, in this project we were limited to 224x224 images. This may not seem like a big difference, but this much of a loss of information and quality of the image can be meaningful for a CNN. For this reason, it was decided to take advantage of the current setup for tests and try different model architectures that do not present this problem. Two of these architectures are the ones we see next.

ResNet-152 [HZRS15] has the advantage that it does not have any additional FC layers other than the output layer, unlike VGG-16. This means that the weights that are learned when pre-training it on ImageNet do not depend on image size, because it is fully convolutional. As a result, any kind of input size can be applied to it. Knowing the state-of-the-art best works on Deep learning, it is expected that ResNet-152 performs better than VGG-16.

Initially the same parameters as specified in the original paper are used to test this architecture, both for the ingredient and the category networks. Given that the results obtained were not as good as expected, some further tests were run on the category network alone.

The procedure followed for these tests is the same as with the previous ones, where each decision is taken depending on the results obtained until the moment. These tests include:

- Variations of learning rate, from 0.1 to 0.001.
- Changes to the learning rate decay set after 10 epochs normally.
- Removal of weight decay.

Finally, no results were good enough to compare to the VGG-16 ones, so no further testing was done, neither with the ingredient network nor the softmax classifier.

6.1.3 MSMVFA

Recipes5k

Now that all base networks have been trained, we can convert at free will the images to features from different types and scales. This means that we are ready to implement the multi-view and multi-scale aggregation of features. The first thing is to decide which mechanisms to use for the aggregation itself and the feature fusion from different patches of an image. For this, the same methods are chosen as in the original implementation explained on the paper, with the objective to be able to get the same results as in that work when moving onto another dataset. All the methods used are the following:

- Z-score is used as the normalization technique applied to the features before both steps of aggregation.
- Simple concatenation is used for the multi-scale aggregation.
- Simple concatenation is also used for the multi-view aggregation.
- Max-pooling is used to fuse features of the 4 patches of an image in the case of L_2. This means that, for each value of a feature vector, the highest value from the 4 patches is taken as the final output.

The implementation itself is as follows: First, the models that were saved in memory are loaded again and features are computed for images of both L_1 and L_2 scales. The features extracted are the following:

- **F_1:** The output of the last FC layer of the category network, which in this case will have a variable length depending on the amount of neurons used in the custom FC layer.

- F_2**:** The output of the ingredient network, having a length of 240, as it is the amount of different ingredients being classified.
- F_3**:** The output of the category network, having a length of 10 in this case (the 10 different dishes chosen for the toy problem).

Next, multi-scale and multi-view aggregations are implemented using the specified techniques in each case. The implementation is divided into different functions so that the aggregation can be applied to any data split (training, validation or test) and even can be applied partially. For example, we can obtain the features resulting from applying only the multi-scale feature aggregation, or the raw features without any aggregation applied. This is necessary to be able to run different experiments to test the usefulness of the method.

Food-101

6.1.4 Final classification tests

Recipes5k

Once the method has been implemented the last thing left to do is run the final classification using a softmax classifier. Again, like in the case of the base networks, we do not care about the results being exactly like the ones in the literature, because they will not be in this toy problem. Furthermore, in the original work little is mentioned about this softmax classifier aside from its name. We know this softmax classifier is the equivalent to a single FC layer that outputs the final prediction. We take similar decisions to the previously trained networks, playing with the applied dropout as well as the learning rate. The Adam optimizer is used once again for this classification. Because of this being a quick classification process because of the size of the network, 200 epochs are used in this case for training.

Following the results obtained by the original paper, we try to divide the classification tests in two different studies:

- Multi-scale tests: running the softmax classifier training, for each of the feature types separately, of using only L_1 features, only L_2 features, and $L_1 + L_2$ features resulting from the multi-scale aggregation. This adds up to a total of 9 tests.
- Multi-view tests: running the softmax classifier training with all possible combinations of feature types (F_1, F_2, F_3), considering features obtained from using the multi-scale aggregation for each of them separately. This adds up to a total of 7 tests.

All parameters chosen for the softmax classifier are kept the same for all tests on the same study (multi-scale or multi-view), so that the results are more comparable. However, the parameters can be different from one study to the other. A full view of the parameters used

and the results obtained from all these tests can be found in the next chapter.

Food-101

When translating the implemented MSMVFA method from the last stage to this one, big changes had to be made. It is no longer possible to put all dataset into memory to then compute and aggregate the features. The way this was handled was by adding a step to the pipeline of the method, which is the feature generation. Once the ingredient and categorical networks have been trained and before the aggregation is made, all images of the dataset are passed through the trained models and a set of features is created and stored in the computer. This includes all necessary features for the method, including the three different types (F_1, F_2, F_3), the two scales being used (L_1, L_2) and the two data splits available in Food-101 (training, test).

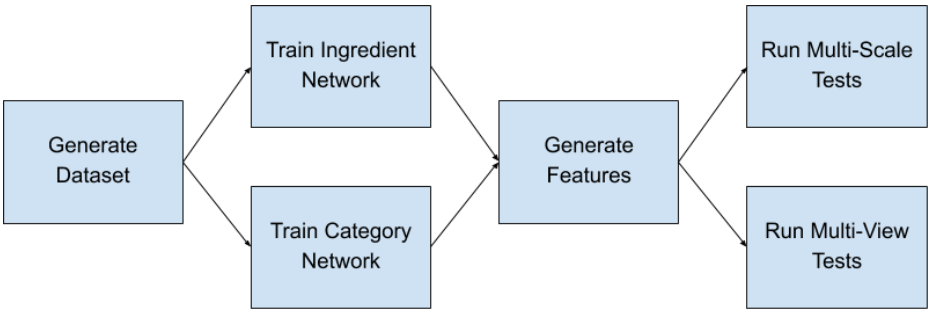

Figure 6.3: Scheme of the code structure to execute to run the whole pipeline

The MSMVFA functions are now modified to adapt to these changes. They are made so that it is possible to obtain any of the necessary combinations of feature aggregation necessary for the multi-scale and multi-view tests. Speaking of which, the tests themselves are also put into easily-accessible functions which will first create the necessary features by aggregating the features stored in disk, and then train the softmax model. Since no information is given about the softmax classifier, the Adam optimizer is still used, and the only changes are the elimination of dropout and the increase in the batch size to adapt to the much larger data-set. Note that, by creating the features earlier, this final classification does not use images or has to load models at all, which increases the training speed considerably.

6.2 Results

In this chapter we will see the results obtained on all different tests done during the development of this project and that we have just seen in the previous chapter. Not only that, but these results will also be discussed and analyzed, specially to better understand

some decision making during the experimental procedure. Final results will naturally be compared to those obtained by the SoA MSMVFA [JMLL19]. The sections ahead are ordered chronologically depending on the time the results were obtained. Only the final and most important results will be shown in this chapter, so for additional results check the Appendix.

6.2.1 Recipes5k

As we saw in **??** the first step was to create a toy problem with the Recipes5k dataset. In this section we will see the results obtained in that set of tests. To begin with, we will take a look at the category and ingredient networks obtained after many different hyperparameter tests. The final ones used are those in Table 6.1, which are used to obtain the results seen in Table 6.2. The metrics used are the accuracy for the category network (single-label problem) while the f1-score is used for the ingredients network (multi-label problem).

Hyperparameter	Category N.	Ingredient N.
Include FCs	No	No
Trainable layers	FC and last 4	FC and last 4
Neurons	32	512
Dropout	0.5	0.5
Optimizer (lr)	Adam(0.0001)	Adam(0.0001)
Epochs	30	30
Batch size	64	64

Table 6.1: Hyperparameters used for the base networks training on Recipes5k

Accuracy	L_1	L_2
Train	79.12	84.56
Val	45.45	40.91
Test	51.25	44.69

F_1	scale 1	scale 2
Train	0.8260	0.7953
Val	0.5128	0.4596
Test	0.5107	0.4503

Table 6.2: Results of accuracy and f1-score obtained on the category and ingredient networks for Recipes5k

We can see that the only difference between the parameters used on one network or the other are the number of neurons in the added FC layer. This is because while there are only 10 food categories in this toy problem, there are more than 200 ingredients to be classified, which asks for a more complex network. On the results shown, we see that the networks are clearly overfitting, which result in test accuracy and f1-score lower than those on the training

set. In both cases we see that the L_2 networks under-perform against their L_1 counterparts. This is expected, as in the category case the network is trying to know which dish is on the image while only seeing a part of it. The same happens with the ingredient network, with the added problem that it could happen that some ingredients are not visible on a specific image patch.

We now move on to the multi-scale and multi-view tests done using a softmax classifier with the characteristics seen in Table 6.3. As we can see, the same hyperparameters are chosen for both tests. With such a small dataset the training of the classifier was done very quickly, which is why some parameters like the batch size can have extreme values. The results obtained for both tests can be found on Table 6.4 and Table 6.5. It is important to note that, because of the nature of the toy problem, the classification results can have a lot of variability, which is why all shown accuracy values are the result of averaging 3 different executions.

Hyperparameter	Multi-scale	Multi-view
Dropout	0,5	0,5
Optimizer (lr)	Adam(0.001)	Adam(0.001)
Epochs	200	200
Batch size	1	1

Table 6.3: Hyperparameters used for the softmax classifier on multi-scale and multi-view tests for Recipes5k

DF	Split	Acc
L_1	Train	80.71
	Val	42.42
	Test	**50.42**
L_2	Train	80.23
	Val	47.47
	Test	**50.83**
$L_1 + L_2$	Train	93.89
	Val	48.99
	Test	**58.33**

MLF	Split	Acc
L_1	Train	69.24
	Val	29.29
	Test	**30.00**
L_2	Train	56.07
	Val	21.72
	Test	**24.58**
$L_1 + L_2$	Train	80.61
	Val	31.82
	Test	**30.83**

HLF	Split	Acc
L_1	Train	53.01
	Val	30.30
	Test	**36.25**
L_2	Train	48.61
	Val	37.88
	Test	**44.17**
$L_1 + L_2$	Train	73.26
	Val	41.92
	Test	**44.58**

Table 6.4: Accuracy results obtained on the multi-scale tests on Recipes5k on the three feature types: deep features (DF), mid-level features (MLF) and high-level features (HLF)

From the multi-scale test results we can observe various things. First, that overfitting is still prevalent in all executions. Second, that deep features are the ones that perform the best in general, followed by high-level features and lastly mid-level features. The latter are features obtained by trying to detect ingredients which are now being used to categorize food, unlike the other two feature types. This explains why they may perform worse. Finally, we observe that in all cases, although using L_1 features only is worse than using L_2 features only or the other way around, using the multi-scale aggregation results in the best performance overall. This proves the usefulness of the method, specially in the deep features case.

Moving on to the second tests, in Table 6.5 we find the results for the 7 different multi-view tests. Using only F_1, F_2 or F_3 should be very similar if not exactly the same as using the $L_1 + L_2$ case on the multi-scale tests. We can see that, indeed, these results are almost identical. What is completely new, however, are the results from aggregating different combinations of feature types. We observe that aggregating either F_2 or F_3 to F_1 does not increase the performance, but when aggregated all 3 together it does. This could be a result of the high variability of the problem, but it can also reflect the usefulness of the multi-view aggregation. In the last case of $F_2 + F_3$ the model is not capable to use the extra ingredient information to its advantage and performs worse than F_3. Overall, however, the best performance is indeed obtained by using all possible features.

	Split	Accuracy
F_1	Train	93.89
	Val	50.00
	Test	**57.92**
F_2	Train	78.22
	Val	28.28
	Test	**28.33**
F_3	Train	72.78
	Val	40.91
	Test	**46.67**

	Split	Accuracy
F_1+F_2	Train	97.04
	Val	46.46
	Test	**57.92**
F_1+F_3	Train	96.94
	Val	44.44
	Test	**56.25**
F_2+F_3	Train	91.60
	Val	38.38
	Test	**42.50**
F_1+F_2+F_3	Train	98.28
	Val	44.44
	Test	**58.33**

Table 6.5: Accuracy results obtained on the multi-view tests on Recipes5k

6.2.2 Food-101

ResNet-152

As explained in **??**, once the experiments with VGG-16 were finished and the overall performance of the models trained was obtained, we began exploring alternative architectures. One of the differences that existed between the original models and the ones developed in this project was the image input size limitation. Because of this, two structures that allowed us to choose a specific input size were chosen.

The first structure is ResNet-152 [HZRS15]. Using it meant having to make some hyperparameter changes. For instance, the batch size had to be changed from 48 to 8 and the learning rate from 0.0001 to 0.001 initially. At first, these hyperparameters were set and both category and ingredient networks were trained. The results obtained can be seen in both Table 6.6 and Figure 6.4. Additionally, in Table 6.6 we see a series of results coming from further testing. Since unsatisfactory results were obtained, an attempt to improve them by modifying hyperparameters was made, similarly to what was done with VGG-16 previously. The experiments are shown in the table in order. First, weight decay was deactivated, which improved the results. From that point, the best learning rate was sought. However, because the best result obtained was not close to the ones we got with VGG-16, the experimentation was stopped here, without getting to the MSMVFA method.

Weight decay	Learning rate	Top-1	Top-5
0.0001	0.001	15.98	31.08
0	0.001	21.73	38.38
0	0.01	27.87	48.81
0	0.1	3.90	14.60
0	0.01 (*)	24.71	43.39
0	0.02	31.16	54.95

Table 6.6: Accuracy results using ResNet-152 as the category network on Food-101 (*) learning rate is divided by 10 at epoch 20 instead of epoch 10

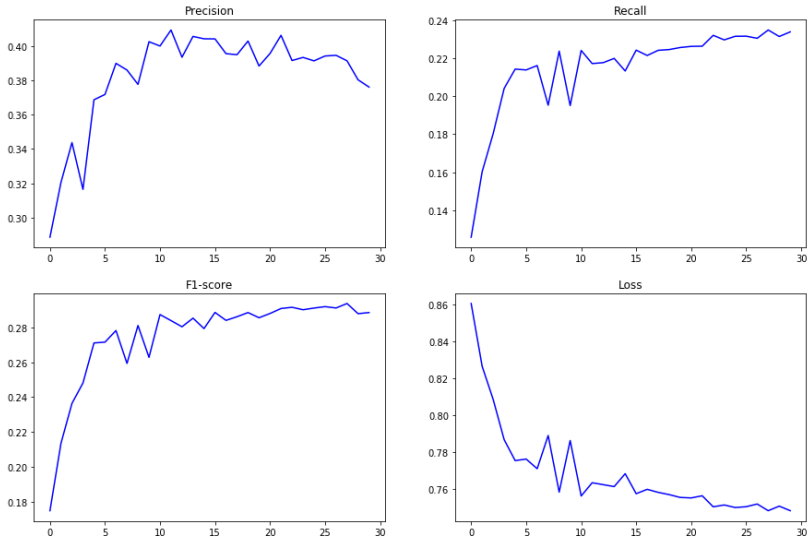

Figure 6.4: Metrics evolution obtained with ResNet-152 as the ingredient network on Food-101

VGG-16

The seen toy problem gives interesting results but, as mentioned before, some of those results could be attributed to the nature of the problem itself. We now move to Food-101, using the whole dataset as done in the original paper. The network training in this case tries to imitate the authors with the hyperparameters seen in Table 6.7. It can be observed that the same parameters are now used for both networks, because of the elimination of the custom FC layer.

Hyperparameter	Category N.	Ingredient N.
Include FCs	Yes	Yes
Trainable layers	FCs	FCs
Dropout	0	0
Optimizer (lr, m)	SGD(0.0001, 0.9)	SGD(0.0001, 0.9)
Epochs	30	30
Batch size	64	64
Weight decay	None	None

Table 6.7: Hyperparameters used for the base networks training on Food-101 for the first time

Accuracy	L_1	L_2
Train	34.41	19.57
Test	35.37	20.56

F_1	L_1	L_2
Train	0.0571	0.0495
Test	0.0563	0.0431

Table 6.8: Results of accuracy and f1-score obtained on the category and ingredient networks for Food-101 for the first time

The results from the network training can be seen on Table 6.8. We notice that there are only training and test sets this time, as Food-101 does not include a validation set. From the category network we can observe that the network has not overfitted at all, leaving all accuracy measures to very low values, for both scales tested. These values are very far from the ones obtained by the original paper, so no comparisons will be made during this stage yet. Taking a look now at the ingredient network we see that the f1-score values obtained are extremely low for all the cases. Compared to what is expected, these results almost mean that the network has not been able to learn properly. For this stage, we move forward with this results knowing that possibly all results related to the ingredient network will be affected.

Next, we move on to the multi-scale and multi-view tests with the softmax classifier. This time, the hyperparameters used are a bit different as seen in Table 6.9. Another thing to note is that different learning rates are used for the training with the deep features than the others. This is because the original paper's learning rate resulted on very weird performances in this case, with the network not being able to learn properly.

Hyperparameter	Multi-scale (DF)	Multi-scale (MLF, HLF)	Multi-view
Dropout	0	0	0
Optimizer (lr)	Adam(0.0001)	Adam(0.001)	Adam(0,0001)
Epochs	200	200	200
Batch size	32	32	32

Table 6.9: Hyperparameters used for the softmax classifier on multi-scale and multi-view tests for the first Food-101 tests

DF	Split	Top-1	Top-5
L_1	Train	50.19	76.69
	Test	**43.34**	**71.13**
L_2	Train	65.21	87.19
	Test	**39.27**	**66.60**
$L_1 + L_2$	Train	76.80	93.84
	Test	**46.05**	**73.17**

MLF	Split	Top-1	Top-5
L_1	Train	30.90	57.57
	Test	**31.97**	**59.88**
L_2	Train	25.96	51.47
	Test	**26.10**	**52.16**
$L_1 + L_2$	Train	37.31	64.29
	Test	**36.98**	**64.79**

HLF	Split	Top-1	Top-5
L_1	Train	36.06	63.06
	Test	**36.45**	**64.12**
L_2	Train	30.63	59.01
	Test	**31.34**	**59.87**
$L_1 + L_2$	Train	41.43	68.71
	Test	**40.81**	**68.15**

Table 6.10: Top-1 and Top-5 accuracy results obtained on the multi-scale tests on Food-101 on the first set of tests for the three feature types

The results for all the multi-scale tests conducted can be seen in Table 6.10. For the DF case, we see that while L_2 performs worse overall, the aggregation is beneficial and results on the best performance. The same can be observed for the other 2 feature types, for both Top-1 and Top-5 accuracy values. The overall performance in all tests keeps being low, which is understandable given that the features learned through the base networks were also performing poorly. However, these results do seem to showcase the usefulness of the multi-scale aggregation in merging information from different granularities and ending up

with a better performance overall.

Finally, we will take a look at the results from the multi-view tests run for Food-101, which are summarized in Table 6.11. Once again, the results for $F_{1,\ 2}$ and F_3 should be equivalent to the previous $L_1 + L_2$ results for each feature type. This is true overall for the values, but we notice a meaningful difference between F_2 and the multi-scale results of MLF. This may be indicative of the variability of the results coming from the features learned from ingredient detection. However, another important difference between the two results is the amount of normalizations applied. On multi-scale tests, features from different scales are normalized and then concatenated together. In multi-view tests, this same procedure is taken but, additionally, the whole feature vector from a specific type is normalized again. This extra normalization step could modify the end results slightly. Overall, the same order as in Recipes5k is kept as from which feature types obtain the higher performance, with F_1 being the best and F_2 the worst. For the other tests, which include features from more than one type, we observe that F_1 is only to obtain better results for Top-1 accuracy when paired with F_3, and the aggregation of F_2 with F_3 brings better results than any of them separately. This suggests that deep features are enough to obtain top-performance, while the more specific information coming from food and ingredient categorization complement each other well.

	Split	Top-1	Top-5
F_1	Train	76.59	93.80
	Test	**45.96**	**73.09**
F_2	Train	32.88	59.49
	Test	**33.99**	**61.8**
F_3	Train	42.49	70.57
	Test	**41.23**	**68.95**
F_1+F_2	Train	76.84	93.84
	Test	**45.69**	**72.83**
F_1+F_3	Train	79.51	95.42
	Test	**46.38**	**72.19**
F2+F_3	Train	45.40	73.36
	Test	**42.93**	**70.51**
F_1+F2+F_3	Train	79.64	95.56
	Test	**45.04**	**71.66**

Table 6.11: Top-1 and Top-5 accuracy results obtained on the multi-view tests on the first tests on Food-101

The first attempt at reproducing the original paper's results obtained metric values far away from ones comparable to them. In this second iteration of results, many hyperparameter values were fixed in order to better align with those used by the authors of the paper, without considering its increasing cost. The final hyperparameters used can be found in Table 6.12. The addition of weight decay is meant to reduce to some extent the overfitting of the network, while the training all the network with a smaller batch size should result in a better performance overall. The different learning rates were a result of trying out different variations. The results for both category and ingredient network using the original learning rate of 0.0001 were not as good as expected, so a little of parameter exploration was done to obtain a better version. The results we want to reproduce are the ones from the multi-scale and multi-view tests, so it is better to part from the best base performance as possible.

In this stage, as we saw in **??**, a new execution environment was used in order to train the category and ingredient networks. This partially was done to solve the problems found in the last iteration of results with the training of the ingredient network, which was attributed to the developed code. The change of environment meant there is only access to the test results, which is why the results in Table 6.13 do not contain any training set metrics. Looking at

the numbers obtained, we can see a substantial improvement compared to the last iteration of results. The category network obtains very good accuracy values, and the f1-score for the ingredient network in the range of expected values. We keep seeing how the networks for scale L_2 perform way worse, like in past tests.

Hyperparameter	Category N.	Ingredient N.
Include FCs	Yes	Yes
Trainable layers	All	All
Dropout	0	0
Optimizer (lr, m)	SGD(0.001, 0.9)	SGD(0.005, 0.9)
Epochs	30	30
Batch size	48	48
Weight decay	0.0001	0.0001

Table 6.12: Hyperparameters used for the base networks training on Food-101 for the second time

Test Metrics	L_1	L_2
Category Top-1 Acc	67.68	48.31
Category Top-5 Acc	88.73	72.74
Ingredient F1-score	0.6387	0.4298

Table 6.13: Results of accuracy and f1-score obtained on the category and ingredient networks for Food-101 for the second time

Once the softmax classifier training procedure starts, we want to make sure that the methodology is as close as the original, because the multi-scale and multi-view tests are the ones that we really want to reproduce. The hyperparameters chosen in Table 6.14 have seen some modifications from the last iteration, given that we know more information about the original method. The number of epochs is kept high because the cost of these executions is pretty low and there have been no cases of accuracy values going down over time.

Hyperparameter	Multi-scale	Multi-view
Dropout	0	0
Optimizer (lr, m)	SGD(0.0001, 0.9)	SGD(0.0001, 0.9)
Epochs	200	200
Batch size	32	32

Table 6.14: Hyperparameters used for the softmax classifier on multi-scale and multi-view tests for the second Food-101 tests

DF	Split	Top-1	Top-5
L_1	Train	99.14	99.93
	Test	**70.82**	**90.43**
L_2	Train	100	100
	Test	**76.87**	**93.43**
$L_1 + L_2$	Train	100	100
	Test	**79.39**	**94.66**

MLF	Split	Top-1	Top-5
L_1	Train	75.94	90.86
	Test	**64.27**	**82.99**
L_2	Train	69.38	89.70
	Test	**63.11**	**85.24**
$L_1 + L_2$	Train	80.75	94.70
	Test	**69.52**	**87.91**

HLF	Split	Top-1	Top-5
L_1	Train	89.53	94.36
	Test	**69.30**	**80.59**
L_2	Train	90.62	99.69
	Test	**62.67**	**88.51**
$L_1 + L_2$	Train	97.40	99.90
	Test	**74.08**	**90.16**

Table 6.15: Top-1 and Top-5 accuracy results obtained on the multi-scale tests on Food-101 on the second set of tests for the three feature types

The multi-scale test results visible in Table 6.15 tell us a very similar story to what we have been seeing so far. The L_2 versions of the features generally perform worse than their L_1 counterparts, although in this case DF shows an exception to this, and by a large margin. This could be due to the deep visual features being able to better capitalize on the image patches and, with their information being fused via max-pooling, obtain a better overall discriminative power. Further than that, we again see that the versions with aggregated features from both scales perform better in all cases, for both top-1 and top-5 accuracy

metrics. This shows once again that the multi-scale aggregation is able to add up discriminative power from different scales and granularities that complement each other. We will now compare these results to the ones obtained by the authors.

DF	Top-1	Top-5
L_1	78.76	94.19
L_2	84.73	96.47
$L_1 + L_2$	83.26	96.02

MLF	Top-1	Top-5
L_1	77.67	88.65
L_2	76.32	91.32
$L_1 + L_2$	78.50	90.06

HLF	Top-1	Top-5
L_1	78.38	94.07
L_2	81.08	95.38
L_1+L_2	84.37	96.38

Table 6.16: Accuracy values obtained by the authors of the original paper on the multi-scale test for Food-101 and VGG-16 [JMLL19]

As seen on Table 6.16, the authors only give results for the test set. We can see that the results are generally a lot better on their version, which means that we have not been able to exactly reproduce them. However, specially for the DF case, $L_1 + L_2$ tests get closer to their values, meaning they are a meaningful improvement to the model. Only a 4% and less than a 2% on top-1 and top-5 accuracy respectively separates our results from those obtained by them on $L_1 + L_2$ of DF. Aside from the raw values, the general rule of the aggregation being better than the singular cases is maintained. Another difference we notice is that in their results L_2 usually performs better than L_1. This could mean that there is a piece of information missing regarding the creation of the L_2 images or the training of the network. The only two differences between our model and theirs that we know of are: their VGG-16 network was implemented using the Caffe platform, and their input image size was of 250x250. These two factors could also be responsible partially of the differences we are noticing.

Next, we take a look at the results from the multi-view test made at this point, seen in Table 6.17. Compared to using the feature types separately, we see that none of the combinations of features is able to surpass the performance of using F_1 alone. Aggregating F_1 with F_2 brings very similar results, while doing it with F_3, as well as all of them together, drops the performance a bit. This seems to imply that the information coming from the ingredients completes the one of the deep visual features better than those from food categorization, which makes sense. Furthermore, we see that the F_2 and F_3 features do complete it each other, giving discriminative information from different points of view, and obtain a better performance when aggregated together. We will now once again compare these results obtained with the ones from the original MSMVFA. Keep in mind that the multi-view tests are made with all available scales for each feature type, which means that the results of the original MSMVFA will include a third scale L_3 which we do not. While the direct values will not be comparable, we want to see whether if the intuition behind what is better is

preserved.

	Split	Top-1	Top-5
F_1	Train	100	100
	Test	**79.60**	**94.50**
F_2	Train	80.77	94.72
	Test	**69.39**	**87.96**
F_3	Train	97.36	99.90
	Test	**74.00**	**90.09**
F_1+F_2	Train	100	100
	Test	**79.53**	**94.44**
F_1+F_3	Train	100	100
	Test	**78.84**	**94.10**
F_2+F_3	Train	98.46	99.97
	Test	**75.52**	**91 01**
$F_1+F_2+F_3$	Train	100	100
	Test	**78.91**	**94.11**

Table 6.17: Top-1 and Top-5 accuracy results obtained on the multi-view tests on the second tests on Food-101

In Table 6.18 we see again that the results from the original MSMVFA paper only contain the test metrics. We see that in their case, even if it is by a very small margin, the best results for both top-1 and top-5 accuracy values are obtained when aggregating all feature types together. All other combinations that demonstrate very similar performance are those that contain F_1 in them. On one hand, this aligns with our results, in which F_1 is by far the best performing feature type. On the other hand, the results show improvement over aggregation, which is not the case for ours. Another element to note is that F_2 is seen to be the worst performing set of features, just like in our results, although in this case it is not able to improve the performance of F_3 when aggregated together. All these differences, again, may be influenced by the fact that their results include 3 scales instead of 2, or by all the method differences we have already mentioned. Other possible factors could be the use of some specific data-augmentation or data-preprocessing which is not mentioned on the paper. Overall, some performance intuitions are kept while others are not, so we can not say that the results have been correctly replicated.

	Top-1	Top-5
F_1	85.89	96.97
F_2	78.60	90.36
F_3	84.94	96.68
F_1+F_2	87.66	97.43
F_1+F_3	87.41	97.33
F_2+F_3	84.30	95.88
$F_1+F_2+F_3$	87.68	97.45

Table 6.18: Accuracy values obtained by the authors of the original paper on the multi-view test for Food-101 and VGG-16 [JMLL19]

6.3 Discussion

In this section we will take an overall look at the performance of the multi-scale multi-view method. Starting with the multi-scale tests conducted, the feature aggregation has proved useful consistently, obtaining better results when incorporating information from different scales than with any of those scales individually. This can be attributed to the information being complementary and not overlapping itself. The use of the max-pooling when fusing the predictions from 4 different patches into one means that the resulting feature vector will be very different from the one coming from using whole images. Using smaller scales results in information that is heavily focusing on the patches individually, more fine-grained. Images from food categories that don't present a specific spatial layout can heavily benefit from this type of information. For example, in a spaghetti dish, which has a very similar appearance everywhere in the image, the finer scales will be able to obtain general information from a different granularity level which complements the general classification. In other cases, however, like with spaghetti with meatballs, if no meatballs are found in some of the patches the classification for that patch will be incorrect, which is why the L_2 category network performs poorly on its own. The use of the max-pooling in this case is useful, because even if meatballs are only present in one of the patches, the classification can be done correctly, although still with less certainty than when using the whole image. Because of this, in many cases the L_2 executions perform worse than the L_1 counterparts.

From the multi-view tests we have seen that the aggregation is not always beneficial. In rare cases it is the reason for a significant drop in performance, but the fact that the results stay the same with added complexity means that it is worse. Using multi-view features is challenging because the features are coming from very different backgrounds. In most cases we have seen, the deep visual features are the ones performing the best. When added with

other features, sometimes it seems like it fails to correctly adapt to the new inclusion. In this regard, the z-score normalization used may not be enough. Another thing to consider is the added complexity. Deep visual features can be a very large vector (2048 or 4096 with ResNet-152 or VGG-16 respectively). When adding 100 new features coming from the dishes, it is logical that the performance is not able to change a lot. Furthermore, many of these softmax classifiers were completely overfitted. Adding more complexity to the network by adding more features means that it may overfit harder, which can be a reason for worse results. However, all this being said, in many cases the multi-view aggregation has proven useful, specially when combining features from very different backgrounds. Ingredient and dish information seem to complement each other very well, gaining better results in many cases. This proves that the method itself can be very useful in the right circumstances, with the right set of features selected.

Chapter 7

Conclusions and Future Lines

In this project we have taken a look at the field of food image analysis as a whole, and more specifically to food recognition. The state-of-the-art multi-scale multi-view feature aggregation method has been introduced and extensively explored in two different datasets and with various configurations. In this chapter we will first take a look at the conclusions we can extract from the development of the project, followed by possible future work that could be done using this thesis as starting point.

The challenges that food recognition poses asks for methods to adapt by finding clever solutions. High intra-class variability and small inter-class variability are only some of the most challenging problems. The used method is able to overcome many of those challenges by using features from different scales and origins succesfully.

As a whole, we have seen that the multi-scale multi-view method is able to obtain way better performances than those obtained by single CNNs. Both stages of the process have been proven to have potential in improving state-of-the art performances in other methods. Because of this being a general framework, a lot of variations could be applied to the method, like different CNN structures for the base networks, or different aggregation and normalization procedures. This also means that it is easily applicable to other existing methods. It could be possible to do a multi-scale multi-view feature aggregation of where the different feature types come from different approaches in the food recognition field.

The work done in this project could be extended in several areas. For one, exploration of other techniques for the generation of different scale patches, which doesn't rely so heavily on the image structure. The use of other networks for the purpose of classifying ingredients and dishes could also be explored. An example of this could be EfficientNet, which should be able to obtain better results on the different given scales, by using specialized networks for each of them. Last but not least, extend even more the work done by adding more scales to the tests or trying out different classifiers other than the softmax layer.

References

[Ase20a] Marc Asenjo. Msmvfa code. https://github.com/Kittus/msmv, 2020.

[Ase20b] Marc Asenjo. Msmvfa on recipes5k. https://colab.research.google.com/drive/1aaL5xYmAGsY-j3TxkpgAsJHG6XF6rP1G, 2020.

[AYB+18] Somak Aditya, Yezhou Yang, Chitta Baral, Yiannis Aloimonos, and Cornelia Fermüller. Image understanding using vision and reasoning through scene description graph. *Computer Vision and Image Understanding*, 173:33 – 45, 2018.

[BFR17] Marc Bolaños, Aina Ferrà, and Petia Radeva. Food ingredients recognition through multi-label learning. In Sebastiano Battiato, Giovanni Maria Farinella, Marco Leo, and Giovanni Gallo, editors, *New Trends in Image Analysis and Processing – ICIAP 2017*, pages 394–402, Cham, 2017. Springer International Publishing.

[BGVG14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[Bol20] Marc Bolaños. Multimodal keras wrapper. https://github.com/MarcBS/multimodal_keras_wrapper, 2020.

[CWWG19] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks, 2019.

[CXH+19] Tianshui Chen, Muxin Xu, Xiaolu Hui, Hefeng Wu, and Liang Lin. Learning semantic-specific graph representation for multi-label image recognition, 2019.

[CZZD17] Xin Chen, Hua Zhou, Yu Zhu, and Liang Diao. Chinesefoodnet: A large-scale image dataset for chinese food recognition. *arXiv preprint arXiv:1705.02743*, 2017.

[DMM19] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep convnet for multi-label classification with partial labels. *CoRR*, abs/1902.09720, 2019.

[ETH20] ETH. Food-101. https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/, 2020.

[Gro20] Vireo Video Retrieval Group. Vireofood-172. http://vireo.cs.cityu.edu.hk/VireoFood172/, 2020.

[HLvdMW16] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[JjC16] Chong-wah NGO Jing-jing Chen. Deep-based ingredient recognition for cooking recipe retrival. *ACM Multimedia*, 2016.

[JMLL19] Shuqiang Jiang, Weiqing Min, Linhu Liu, and Zhengdong Luo. Multi-scale multi-view deep feature aggregation for food recognition. *IEEE Transactions on Image Processing*, PP:1–1, 07 2019.

[KAO14] Hokuto Kagaya, Kiyoharu Aizawa, and Makoto Ogawa. Food detection and recognition using convolutional neural network. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 1085–1088, New York, NY, USA, 2014. Association for Computing Machinery.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[LFH16] Chao Li, Zhiyong Feng, and Yahong Han. Image attribute learning with ontology guided fused lasso. *Multimedia Tools Appl.*, 75(12):7029–7043, June 2016.

[Lu16] Yuzhen Lu. Food image recognition by using convolutional neural networks (cnns), 2016.

[Mid20] Midea. Chinesefoodnet. https://sites.google.com/view/chinesefoodnet/, 2020.

[MJR+15] A. Myers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy. Im2calories: Towards an automated mobile vision food diary. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1233–1241, 2015.

[MJS+17] W. Min, S. Jiang, J. Sang, H. Wang, X. Liu, and L. Herranz. Being a supercook: Joint food attributes and multimodal content modeling for recipe retrieval and exploration. *IEEE Transactions on Multimedia*, 19(5):1100–1113, 2017.

[SLJ+14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[Tea20] Keras Team. Keras applications. https://keras.io/applications/, 2020.

[UB20a] UB. Ingredients-101. http://www.ub.edu/cvub/ingredients101/, 2020.

[UB20b] UB. Recipes5k. http://www.ub.edu/cvub/recipes5k/, 2020.

[ZL16] F. Zhou and Y. Lin. Fine-grained image classification by exploring bipartite-graph labels. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1124–1133, 2016.

[ZMZF19] Ji Zhang, Kuizhi Mei, Yu Zheng, and Jianping Fan. Learning multi-layer coarse-to-fine representations for large-scale image classification. *Pattern Recognition*, 91:175 – 189, 2019.

[ZQL+19] Yan Zhang, Yanyun Qu, Cuihua Li, Yunqi Lei, and Jianping Fan. Ontology-driven hierarchical sparse coding for large-scale image classification. *Neurocomputing*, 360:209 – 219, 2019.

[ZSXL19] X. Zhang, X. Sun, C. Xie, and B. Lun. From vision to content: Construction of domain-specific multi-modal knowledge graph. *IEEE Access*, 7:108278–108294, 2019.